4. Integer Linear Programming

Introduction

An Integer Linear Programming problem is an optimisation problem in the form:

$$\begin{aligned} \min \quad & \underline{c}^T \underline{x} \\ \text{s.t.} \quad & A\underline{x} \geq \underline{b} \\ & \underline{x} \geq \underline{0}, \quad \underline{x} \in \mathbb{Z}^n \end{aligned}$$

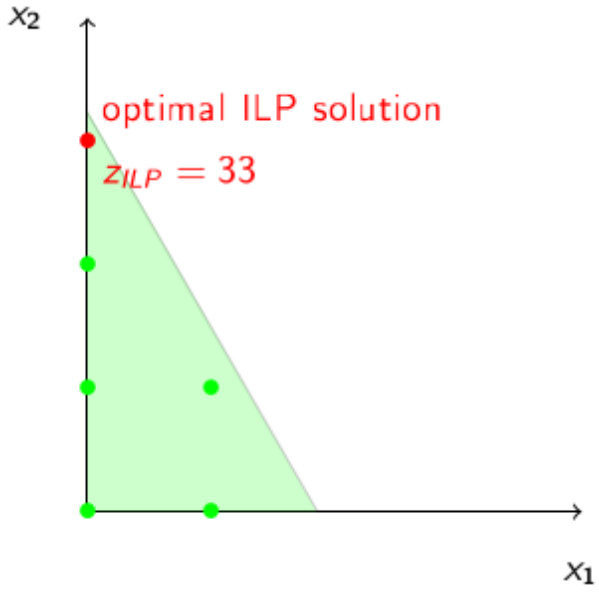
We can also say that:

- If $x_j \in \{0, 1\},\ \forall j$, we have a **binary** Linear Programming problem
- If not all x_j are integer, we have a **mixed** Linear Programming problem

> **Assumption**
>
> We will assume that the parameters $A, \underline{b}$ are integer, without loss of generality.

If we take a look at an example problem, we can see that the main difference between a Linear Programming and an Integer Linear Programming problem is that, in the Integer Linear Programming problem, the feasible region is actually a **lattice**, not a continuous area:

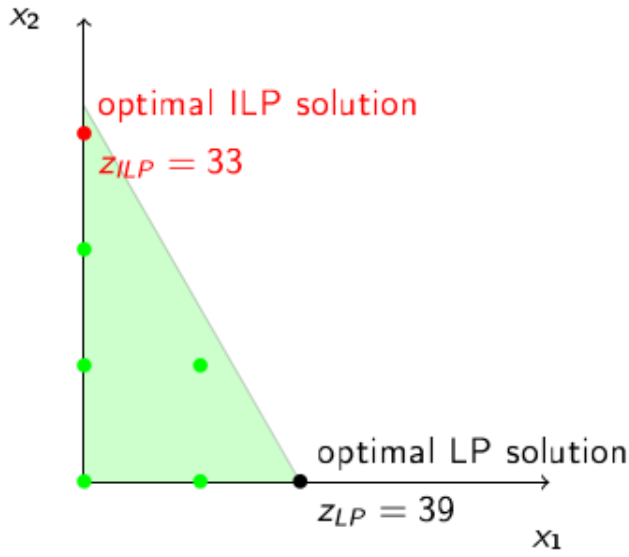
$$\begin{aligned} z_{ILP} = \max \quad & z = 21x_1 + 11x_2 \\ \text{s.t.} \quad & 7x_1 + 4x_2 \leq 13 \\ & x_1, x_2 \geq 0 \text{ integer} \end{aligned}$$



If we were to delete integrality constraints, we would obtain a completely different solution:

Deleting the integrality constraints we obtain the Linear Program:

$$z_{LP} = \max \quad z = 21x_1 + 11x_2$$
$$\text{s.t.} \quad 7x_1 + 4x_2 \leq 13$$
$$x_1, x_2 \geq 0$$



Removing the integrality constraints from an Integer Linear Programming problem leads us to find the so called **linear relaxation** of said problem.

Property 1

For any maximisation Integer Linear Programming problem, we have $z_{\text{ILP}} \leq z_{\text{LP}}$; that is, z_{LP} is an **upper bound** on the optimal value of the corresponding Integer Linear Programming problem.
This also works the other way around for minimisation problems.

One simple way we could use to find Integer Linear Programming problem solutions could be to actually use the linear relaxation of the problem, find the optimal solution in the real space, and then approximate that solution to an integer.

Unfortunately, the rounded optimal solutions of Linear Programming problems are sometimes infeasible and sometimes useless for Integer Linear Programming.

We can see this through the **knapsack problem**:

Knapsack problem

Given:

1. n objects $j = 1, \ldots, n$
2. p_j value of object j
3. w_j weight of object j
4. b maximum knapsack capacity

Determine a subset of objects that maximises total profit, while respecting the knapsack's capacity.

It has been proven that the binary knapsack problem is NP-hard. However, the binary knapsack problem has a wide range of direct and indirect applications, such as:

- Loading of containers or vehicles
- Investments

There are also several extensions of the knapsack problem, among which:

- Integer knapsack problem (with multiple copies of each object)
- Multi-dimensional knapsack (with at least two capacity constraints)

Let's take a look at some more Integer Linear Programming problems:

Assignment problem

Given:

1. m machines, $i = 1, \ldots, m$
2. n jobs, $j = 1, \ldots, n$
3. c_{ij} cost of assigning job j to machine i

Determine an assignment of jobs to the machines so as to minimise the total cost, while assigning at least one job per machine and at most one machine for each job.

Transportation problem

Given:

1. m production plants, $i = 1, \ldots, m$
2. n clients, $j = 1, \ldots, n$
3. c_{ij} transportation cost of one unit of product from plant i to client j
4. p_i production capacity of plant i
5. d_j demand of client j
6. q_{ij} maximum amount to be transported from plant i to client j

Determine a transportation plan that minimises total costs while satisfying plant capacity and client demands.

Assumption

$$\sum_{i=1}^{m} p_i \geq \sum_{j=1}^{n} d_j$$

Regarding the above problems, we know that the following property is true:

Property of transportation and assignment problems

For both transportation and assignment problems, the optimal solution of their linear relaxation coincides with the optimal solution of the original problem

For this reason, we have polynomial time algorithms to solve both problems easily and efficiently.

The property above is true because the following theorem is correct:

Theorem 3

If, in a transportation problem, p_i, d_j, q_{ij} are all integer, all the basic feasible solutions of its linear relaxation are also integer.

This theorem has a couple of consequences that are directly visible in the form of the problem:

- The integer constraint matrix A has elements with values $-1, 0, 1$ only and exactly three non-zero coefficients per column
- The $\underline{b}$ vector only has integer values

The optimal solution of the linear relaxation of a problem with this form is:

$$\underline{x}^* = \begin{bmatrix} B^{-1}\underline{b} \\ \underline{0} \end{bmatrix}$$

Where the inverted B matrix has the form:

$$B^{-1} = \frac{1}{|B|} \begin{bmatrix} \alpha_{11} & \cdots & \alpha_{1n} \\ \vdots & \ddots & \vdots \\ \alpha_{m1} & \cdots & \alpha_{mn} \end{bmatrix}$$

Where $\alpha_{ij} = (-1)^{i+j}|M_{ij}|$, with M_{ij} being the submatrix obtained from B by eliminating row i and column j.

Knowing these facts, we can safely say that:

- B integer $\implies$ α_{ij} integer
- $|B| = \pm 1 \implies B^{-1}$ integer $\implies \underline{x}^*$ integer

It can be proven that A is totally **unimodular**, which means that $|Q| \in \{-1, 0, 1\}$ for any square submatrix Q of A.

Scheduling problem

Given:

1. m machines, $k = 1, \ldots, m$
2. n jobs, $j = 1, \ldots, n$
3. d_j deadline for job j, $j = 1, \ldots, n$
4. p_{jk} processing time of job j on machine k (may be nil)

Assumption

Each job must be processed once on each machine following the order of the machine indices.

Determine an optimal sequence in which to process the jobs so as to minimise the total completion time while satisfying the deadlines.

As we can tell, most Integer Linear Programming problems are NP-hard. There are no efficient algorithms to solve them, and the existence of a polynomial time algorithm for any one of them would imply that $P = NP$, which is extremely unlikely.

Some methods used to solve these problems are:

- Implicit enumeration (exact)
- Cutting planes (exact)
- Heuristic algorithms (approximate)

As an example, we list some implicit enumeration methods below:

- Branch and bound method
- Dynamic Programming method

4.1. Branch and bound method

Consider a generic optimisation problem:

$$\min\{c(\underline{x}) : \underline{x} \in X\}$$

The main idea behind the branch and bound method is to reduce the solution of a difficult problem to that of a sequence of simpler subproblems by partitioning the feasible region X.

The algorithm consists of two phases, as the name suggests:

1. **Branching**: partition X into k subsets:

$$X = X_1 \cup \ldots \cup X_k,\ X_i \cap X_j = \emptyset\ \forall i \neq j$$

and let $z_i = \min\{c(\underline{x}) : \underline{x} \in X_i\}$, for $i = 1, \ldots, k$. It is clear that:

$$z = \min\{c(\underline{x}) : \underline{x} \in X\} = \min\{z_1, \ldots, z_k\}$$

2. **Bounding**: for each subproblem z_i, either:

   - determine an optimal solution;
   - prove that the problem is infeasible;
   - prove that $z_i \geq z'$, which in turn is the objective function value of the best feasible solution found so far.

   If the problem cannot be "solved" in any of these three ways, we further generate new subproblems and try to solve those.

4.1.1. Branch and bound for Integer Linear Programming

Given an Integer Linear Programming problem:

$$\min\{\underline{c}^T \underline{x} : A\underline{x} = \underline{b}, \underline{x} \geq \underline{0}, \underline{x} \in \mathbb{Z}^n\}$$

We can apply the branch and bound algorithm to solve it by first branching:

Branching

Partition X into subregions.

Let $\overline{\underline{x}}$ denote an optimal solution for the linear relaxation of the Integer Linear Programming problem $\min\{\underline{c}^T \underline{x} : A\underline{x} = \underline{b}, \underline{x} \geq \underline{0}\}$ and $z_{\text{LP}} = \underline{c}^T \overline{\underline{x}}$ denote the corresponding optimal value.

If $\overline{\underline{x}}$ is integer, $\overline{\underline{x}}$ is also optimal for the Integer Linear Programming problem. Otherwise, $\exists \overline{x}_h$ (a fractional number) and we consider two subproblems:

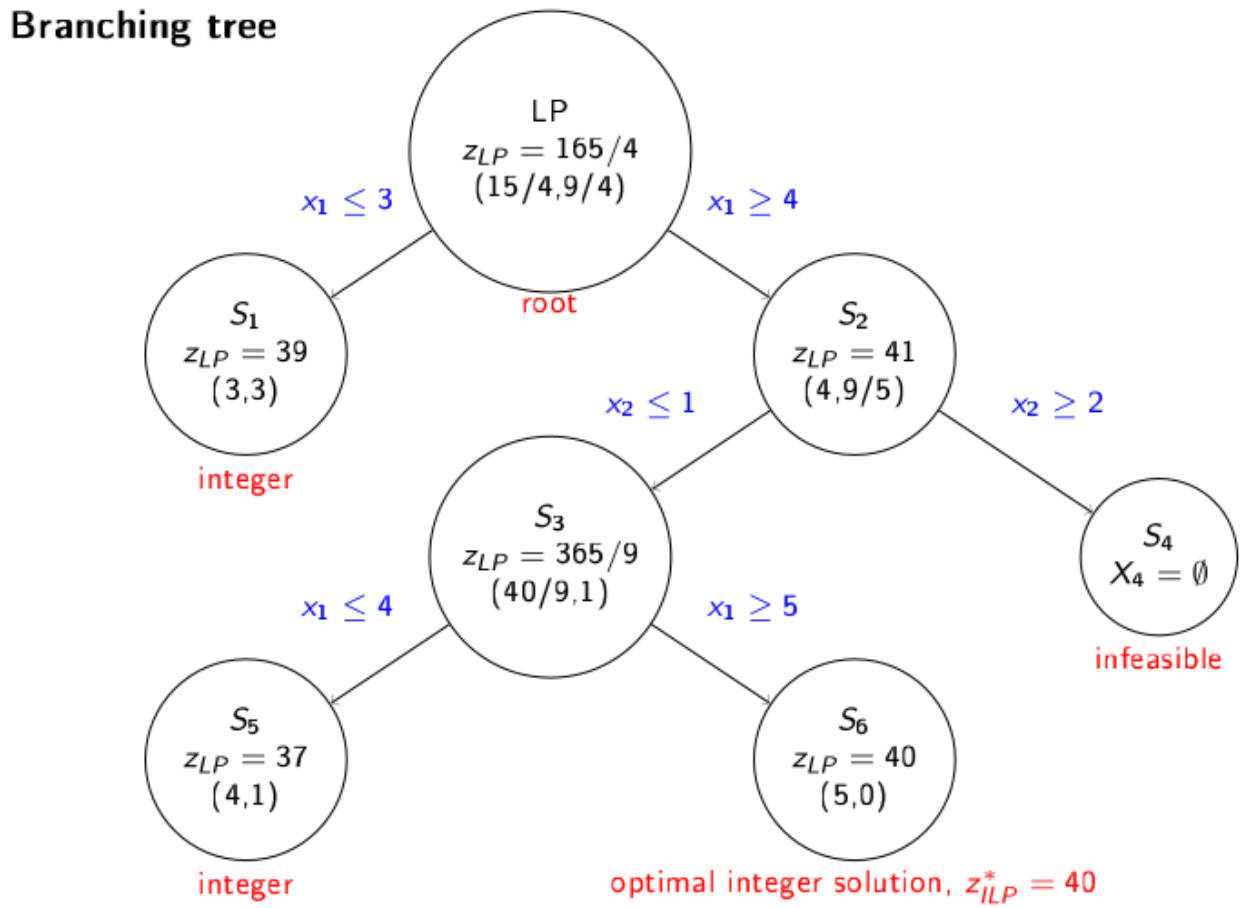
$$\text{ILP}_1 : \min\{\underline{c}^T \underline{x} : A\underline{x} = \underline{b}, x_h \leq \lfloor \overline{x}_h \rfloor, \underline{x} \geq \underline{0}, \underline{x} \in \mathbb{Z}^n\}$$
$$\text{ILP}_2 : \min\{\underline{c}^T \underline{x} : A\underline{x} = \underline{b}, x_h \geq \lfloor \overline{x}_h \rfloor + 1, \underline{x} \geq \underline{0}, \underline{x} \in \mathbb{Z}^n\}$$

and then bounding:

Bounding

Determine a lower bound on the optimal value z_i of a subproblem by solving its linear relaxation.

The result of branch and bound for Integer Linear Programming problems is a **branching tree**:



The branching tree may not contain all possible nodes (exactly 2^d leaves). In fact, we say that a node of the tree is **fathomed** (has no child) if:

- Initial constraints, intersected with those on the arcs which bring to the current node, make an empty feasible region
- The optimal solution of the linear relaxation is already an integer
- The value $\underline{c}^T \overline{\underline{x}}_{\text{LP}}$ of the optimal solution of the linear relaxation is worse than that of the best feasible solution of the Integer Linear Programming problem found so far

Note that these three conditions for fathoming are also the same three conditions that we saw in the bounding criterion.

We can decide how to traverse the branching tree as well, when applying the branch and bound algorithm:

- We can visit deeper nodes first (**depth-first search strategy**): it's a simple recursive procedure, but can be costly when choosing the wrong node
- We can visit "more promising" nodes first, that is, visit nodes with the best linear relaxation values (**best-bound first strategy**): this typically generates a smaller number of nodes, but subproblems are less constrained, which in turn means that it takes longer to find a first feasible solution and improve it

We can also choose the best fractional variable to branch more efficiently:

- It may not be the best choice to select the variable x_h whose fractional value is closer to 0.5, hoping to obtain two subproblems that are more stringent and balanced
- **Strong branching**: try to branch on some of the candidate variables, evaluate the corresponding objective function values and branch on the variable that yields the **best improvement** in objective function

For subproblems, there is no need to solve their linear relaxations from scratch with the two-phase Simplex algorithm. Instead, an optimal solution with a single additional constraint can be found via a **single iteration** of the **Dual Simplex method** to the optimal solution of the previous linear relaxation.

ⓘ Remarks on branch and bound

- Branch and bound is also applicable to mixed Integer Linear Programming problems: when branching, only consider the fractional variables that must be integer
- Finding a good initial feasible solution with a heuristic algorithm may improve the method's efficiency by providing a better lower bound z' on z_{ILP} (for maximisation problems)
- Branch and bound can also be used as a heuristic algorithm by limiting the compute time or number of nodes that are examined

All in all, branch and bound is a general method that can be adapted to tackle any discrete optimisation problem and many non-linear optimisation problems. All we need to apply it is:

- A technique to partition a set of feasible solutions into two or more subsets of feasible solutions
- A procedure to determine a bound on the cost of any solution in such a subset of feasible solutions

4.2. Cutting plane methods and Gomory fractional cuts

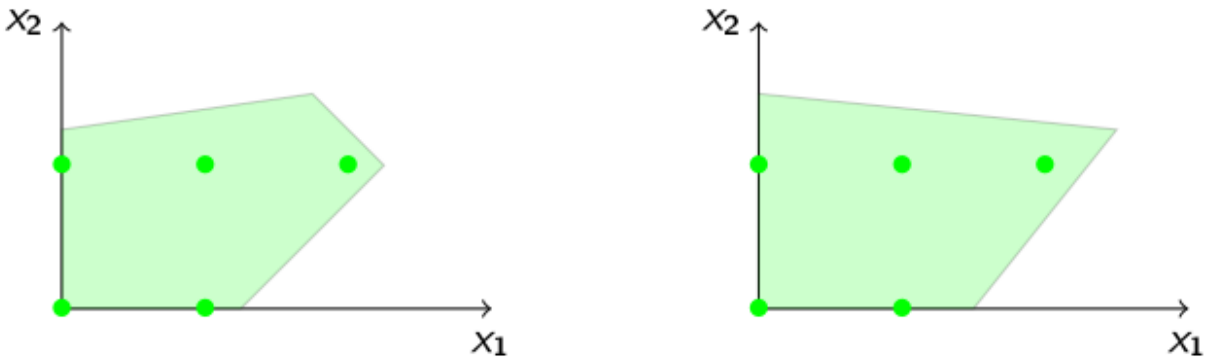
Consider a generic Integer Linear Programming problem:

$$\begin{aligned} \min \quad & \underline{c}^T \underline{x} \\ \text{s.t.} \quad & A\underline{x} \geq \underline{b} \\ & \underline{x} \geq \underline{0}, \quad \underline{x} \in \mathbb{Z}^n \end{aligned}$$

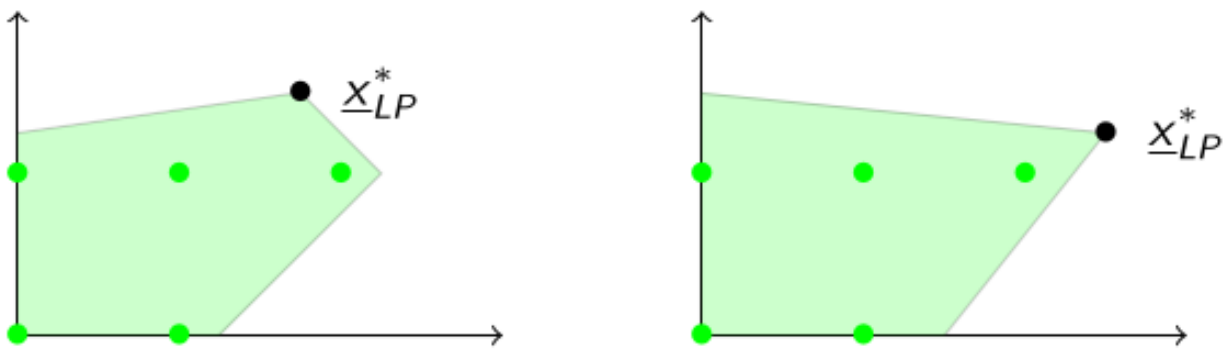
with the feasible region $X = \{\underline{x} \in \mathbb{Z}^n : A\underline{x} \geq \underline{b}, \underline{x} \geq \underline{0}\}$.

Let's assume that a_{ij}, c_j, b_i are all integer coefficients.

The feasible region of an Integer Linear Programming problem can be described by different sets of constraints, which may be weaker:



This means that there are infinitely many formulations for the feasible region of any given Integer Linear Programming problem. However, while all formulations are equivalent, the optimal solutions of their respective linear relaxations can differ substantially:



Ideal formulation

The ideal formulation is that describing the **convex hull** of X, $\text{conv}(X)$, where $\text{conv}(X)$ is the smallest convex subset containing X.

Since all vertices have all integer coordinates, for any $\underline{c}$, we have $z^*_{\text{LP}} = z^*_{\text{ILP}}$ and the Linear Programming optimum is also the Integer Linear Programming optimum.

Theorem

For any feasible region X of an Integer Linear Programming problem (either bounded or unbounded), there exists an ideal formulation (that is, a description of $\text{conv}(X)$ involving a finite number of linear constraints) but the number of constraints can be very large with respect to the size of the original formulation.

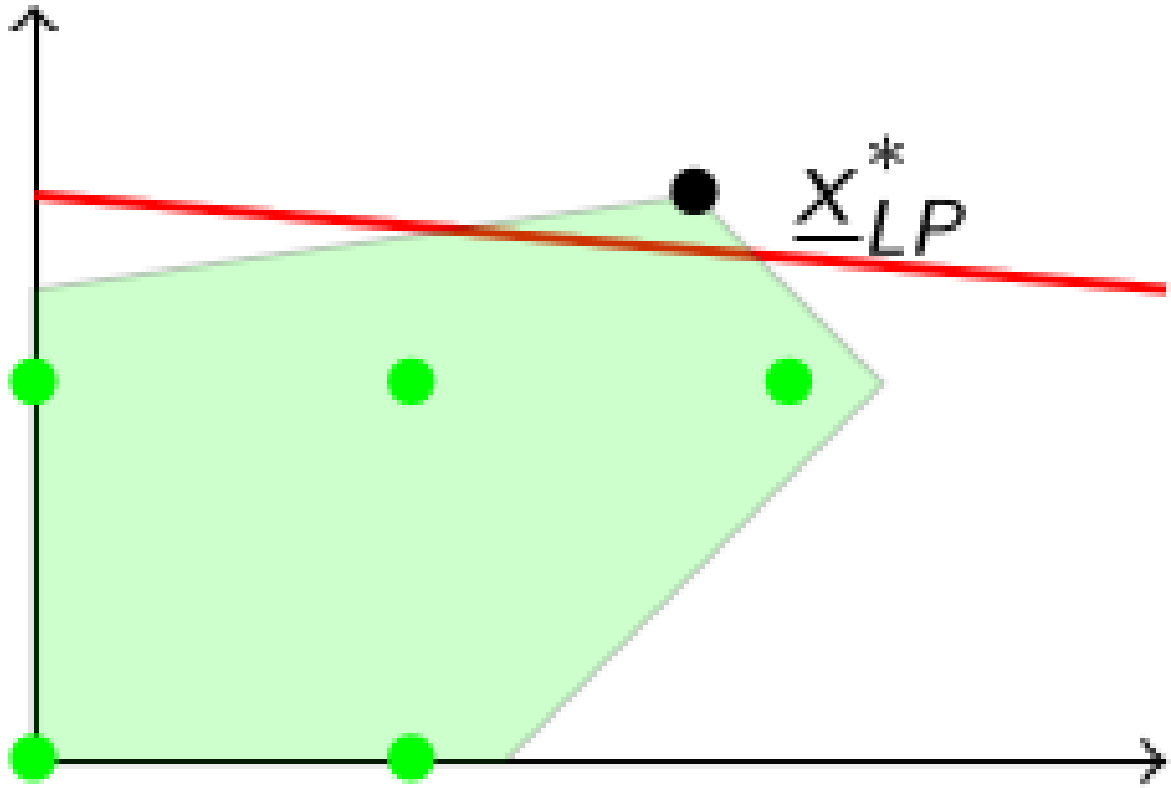
Theoretically, the solution of any Integer Linear Programming problem can be reduced to that of a single Linear Programming problem. However, the ideal formulation is often very large or very difficult to determine.

4.2.1. Cutting plane methods

A full description of $\text{conv}(X)$ is not required, however; we just need a good description in the neighbourhood of the optimal solution.

Cutting plane

A cutting plane is an inequality $\underline{a}^T \underline{x} \leq b$ that is not satisfied by $\underline{x}^*_{\text{LP}}$ but is satisfied by all the feasible solutions of the Integer Linear Programming problem.



The idea is, given an initial formulation, iteratively add cutting planes as long as their linear relaxation does not provide as optimal integer solution.

4.2.2. Gomory fractional cuts

Let $\underline{x}^*_{\text{LP}}$ be an optimal solution of the linear relaxation of the current formulation $\min\{\underline{c}^T\underline{x} : A\underline{x} = \underline{b}, \underline{x} \geq \underline{0}\}$ and $x^*_{B[r]}$ be a fractional basic variable.

The corresponding row of the optimal tableau is:

$$x_{B[r]} + \sum_{j \in N} \bar{a}_{rj} x_j = \bar{b}_r$$

Gomory cut

The Gomory cut with regards to the fractional basic variable $x_{B[r]}$ is:

$$\sum_{j \in N} \left(\bar{a}_{rj} - \lfloor \bar{a}_{rj} \rfloor\right) \geq \left(\bar{b}_r - \lfloor \bar{b}_r \rfloor\right)$$

We can verify that the above inequality is actually a cutting plane with regards to x^*_{LP}.

Firstly, the inequality is not satisfied by the optimal fractional solution $\underline{x}^*_{\text{LP}}$ of the linear relaxation. This is obvious, since $(\bar{b}_r - \lfloor \bar{b}_r \rfloor) > 0$ and $x_j = 0,\ \forall j$ such that x_j is a non-basic variable.

However, the inequality is satisfied by all integer feasible solutions: for each feasible solution of the linear relaxation, we get:

$$x_{B[r]} + \sum_{j \in N} \lfloor \bar{a}_{rj} \rfloor x_j \leq x_{B[r]} + \sum_{j \in N} \bar{a}_{rj} x_j = \bar{b}_r,\ x_j \geq 0$$

In particular, for each *integer* feasible solution, we have:

$$x_{B[r]} + \sum_{j \in N} \lfloor \bar{a}_{rj} \rfloor x_j \leq \lfloor \bar{b}_r \rfloor,\ x_j \in \mathbb{Z}^n$$

By subtracting the above equation from the optimal tableau equation, we obtain:

$$\sum_{j \in N} \left(\bar{a}_{rj} - \lfloor \bar{a}_{rj} \rfloor \right) x_j \geq \bar{b}_r - \lfloor \bar{b}_r \rfloor$$

Now, the integer form and the fractional form of the cutting plane are **equivalent**.

⁝≡ Example

Let's take a look at the following Integer Linear Programming problem:

$$\begin{array}{lrrrrl} \max & z = & 8x_1 & +5x_2 & & \\ \text{s.t.} & & x_1 & +x_2 & \leq 6 & \\ & & 9x_1 & +5x_2 & \leq 45 & \\ & & x_1, & x_2 & \geq 0 & \text{integer} \end{array}$$

The optimal tableau for this problem is:

		x_1	x_2	s_1	s_2
$-z$	-41.25	0	0	-1.25	-0.75
x_1	3.75	1	0	-1.25	0.25
x_2	2.25	0	1	2.25	-0.25

The fractional optimal basic solution relative to the tableau is $\underline{x}^*_B = (3.75, 2.25)^T$.

Now, we select a row of the optimal tableau (that is, a constraint) whose basic variable has a fractional value:

$$x_1 - 1.25s_1 + 0.25s_2 = 3.75$$

Then, we generate the corresponding Gomory cut:

$$0.75s_1 + 0.25s_2 \geq 0.75$$

Now we can introduce the slack variable $s_3 \geq 0$ and add this cutting plane to the tableau. The new constraint "cuts" the optimal fractional solution of the linear relaxation of the Integer Linear Programming problem:

		x_1	x_2	s_1	s_2	s_3
$-z$	-41.25	0	0	-1.25	-0.75	0
x_1	3.75	1	0	-1.25	0.25	0
x_2	2.25	0	1	2.25	-0.25	0
s_3	-0.75	0	0	-0.75	-0.25	1

To efficiently reoptimise the tableau, we can apply a single iteration of the dual simplex algorithm, thus obtaining:

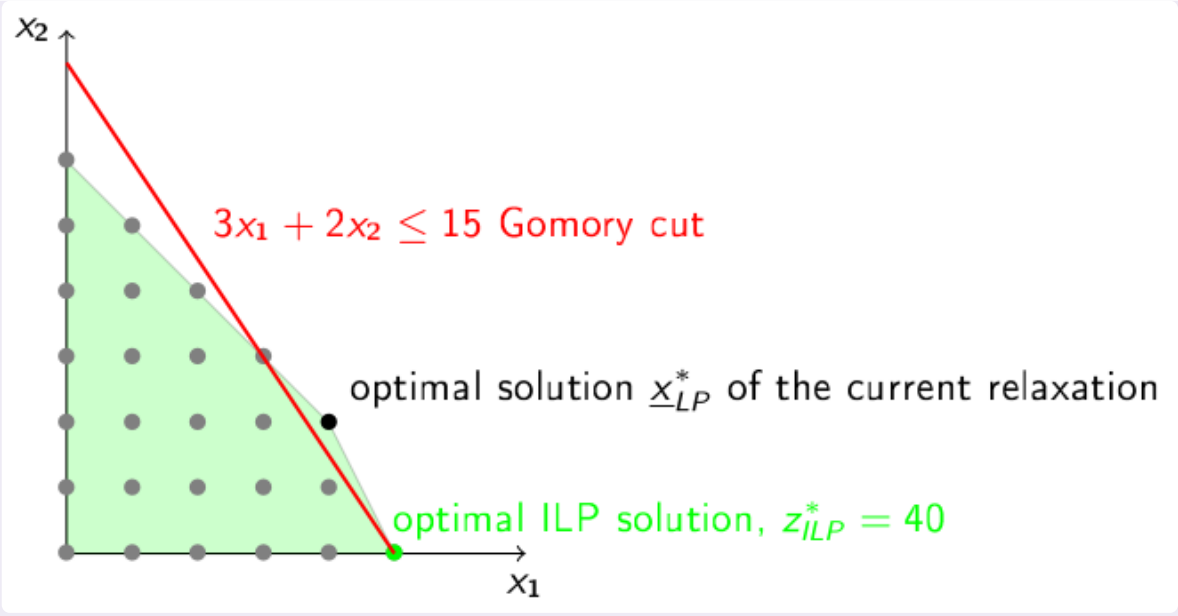
		x_1	x_2	s_1	s_2	s_3
$-z$	-40	0	0	0	-0.33	-1.67
x_1	5	1	0	0	0.67	-1.67
x_2	0	0	1	0	-1	3
s_1	1	0	0	1	0.33	1.33

Since the optimal solution $\underline{x}^* = (5, 0, 1, 0, 0)^T$ of the linear relaxation of the new formulation is integer, $\underline{x}^*$ is **also optimal for the original Integer Linear Programming problem**. We don't need to generate any more Gomory cuts.

To express the Gomory cut $0.75s_1 + 0.25s_2 \geq 0.75$ in terms of the decision variables, we can perform a simple substitution:

$$\begin{cases} s_1 = 6 - x_1 - x_2 \\ s_2 = 45 - 9x_2 - 5x_2 \end{cases} \implies 3x_1 + 2x_2 \leq 15$$

Geometrically, we obtain:



4.2.3. Cutting plane method with fractional Gomory cuts

Let's take a look at the cutting plane method with fractional Gomory cuts:

Cutting plane method with fractional Gomory cuts

1. Solve the linear relaxation $\min\{\underline{c}^T\underline{x} : A\underline{x} = \underline{b}, \underline{x} \geq \underline{0}\}$
2. Let $\underline{x}^*_{\text{LP}}$, be an optimal basic feasible solution
3. While $\underline{x}^*_{\text{LP}}$, has fractional components, do:
    1. Select a basic variable with a fractional value
    2. Generate the corresponding Gomory cut
    3. Add constraint to the optimal tableau of the linear relaxation
    4. Perform one iteration of the dual simplex algorithm

Theorem

If the Integer Linear Programming problem has a finite optimal solution, the cutting plane method finds one after adding a **finite** number of Gomory cuts.

It must be noted that the number of cuts that need to be added is usually very large.

4.2.4. Generic and specific cutting planes

There exist other types of generic cutting planes (which are different from the fractional Gomory cuts) and a large number of classes of cutting planes for specific problems.

The deepest cuts are the facets of $\text{conv}(X)$.

The thorough study of the combinatorial structure of various problems, such as the Travelling Salesman Problem, set covering, set packing and more, led to:

- The characterisation of entire classes of facets
- The efficient procedures for generating them

4.2.5. Idea of Branch-and-Cut

The "combined" **Branch-and-Cut** approach aims to overcome the disadvantages of pure Branch-and-Bound and pure cutting plane methods.

For each subproblem (node) of Branch-and-Bound, several cutting planes are generated to improve the bound and try to find an optimal integer solution. Whenever the cutting planes become less effective, cut generation is stopped and a branching operation is performed.

The advantages of this approach are:

- The cuts tend to strengthen the formulation of the various subproblems
- The long series of cuts without sensible improvement are interrupted by branching operations